Discovering Target-Branched Declare Constraints

Claudio Di Ciccio\textsuperscript{1}, Fabrizio Maria Maggi\textsuperscript{2}, and Jan Mendling\textsuperscript{1}

\textsuperscript{1} Vienna University of Business and Economics, Austria
\{claudio.di.ciccio, jan.mendling\}@wu.ac.at
\textsuperscript{2} University of Tartu, Estonia
f.m.maggi@ut.ee

Abstract. Process discovery is the task of generating models from event logs. Mining processes that operate in an environment of high variability is an ongoing research challenge because various algorithms tend to produce spaghetti-like models. This is particularly the case when procedural models are generated. A promising direction to tackle this challenge is the usage of declarative process modelling languages like Declare, which summarise complex behaviour in a compact set of behavioural constraints. However, Declare constraints with branching are expensive to be calculated. In addition, it is often the case that hundreds of branching Declare constraints are valid for the same log, thus making, again, the discovery results unreadable. In this paper, we address these problems from a theoretical angle. More specifically, we define the class of Target-Branched Declare constraints and investigate the formal properties it exhibits. Furthermore, we present a technique for the efficient discovery of compact Target-Branched Declare models. We discuss the merits of our work through an evaluation based on a prototypical implementation using both artificial and real-world event logs.

Keywords: Process Mining, Discovery, Declarative Processes.

1 Introduction

Process discovery is the important initial step of business process management that aims at arriving at an as-is model of an investigated process \textsuperscript{8}. Due to this step being difficult and time-consuming, various techniques have been proposed to automatically discover a process model from event logs. These log data are often generated from information systems that support parts or the entirety of a process. The result is typically presented as a Petri net or a similar kind of flow chart and the automatic discovery is referred to as process mining.

While process mining has proven to be a power technique for structured and standardised processes, there is an ongoing debate on how processes with a high degree of variability can be effectively mined. One approach to this problem is to generate a declarative process model, which rather shows the constraints of behaviour instead of the available execution sequences. The resulting models
are represented in languages like Declare. In many cases they provide a way to represent complex, unstructured behaviour in a compact way, which would look overly complex in a spaghetti-like Petri net. However, simple branching statements like “if you do a, you will do eventually either b or c” cannot be easily mined for Declare models.

In this paper, we address the problem of mining Declare branching constraints. We define the class of Target-Branched Declare and devise efficient mining algorithms for it. The key idea is to exploit dominance relationships, which help to drastically prune the search space. We present formal proofs to demonstrate its merits. A prototypical implementation is used for performance analysis, emphasising feasibility and efficiency for our approach.

Against this background, this paper is structured as follows. Section 2 introduces the essential concepts of Declare. Section 3 provides the formal foundations for mining Target-Branched constraints. Section 4 defines the construction of a knowledge base from which the final constraint set is built. Section 5 describes the performance evaluation. Section 6 investigates our contribution in the light of related work. Section 7 concludes the paper with an outlook on future research.

2 Background on Mining Declarative Process Models

One of the challenges in process mining is the compact presentation of the mined behaviour. It has been observed that procedural models such as Petri nets tend to become overly complex for flexible processes that are situated in a dynamic environment. Therefore, it has been argued to rather utilise declarative models in such a context, in order to facilitate better understanding of the mined process by humans \[9,22\].

One of the most frequently used declarative languages is Declare introduced by Pesic and van der Aalst in \[26\]. Instead of explicitly specifying the sequence of events, Declare consists of a set of constraints that are applied to activities. Constraints, in turn, are based on templates that define parametrised classes of properties. Templates have a graphical representation and their semantics can be formalised using formal logics \[21,7\], the main one being Linear Temporal Logic over finite traces (LTLf). In this way, analysts work with the graphical representation of templates, while the underlying formulas remain hidden. Table 1 summarises important Declare templates. For a complete specification see \[26\]. Here, we indicate template parameters with \(x\) or \(y\) symbols and real activities in their instantiations with \(a\), \(b\) or \(c\) letters.

The formulas shown in Table 1 can be readily formulated using natural language. The RespondedExistence template specifies that if \(x\) occurs, then \(y\) should also occur (either before or after \(x\)). The Response template specifies that when \(x\) occurs, then \(y\) should eventually occur after \(x\). The Precedence template indicates that \(y\) should occur only if \(x\) has occurred before. The templates AlternateResponse and AlternatePrecedence strengthen the Response and Precedence templates respectively by specifying that activities must alternate without repetitions in between. Even stronger ordering relations are specified by